Apprentissage de GAI-décompositions

Damien Bigot, Hélène Fargier, Jérôme Mengin, Bruno Zanuttini

Apprentissage de GAI-décompositions

Damien Bigot[1], **Hélène Fargier**[1], **Jérôme Mengin**[1], **Bruno Zanuttini**[2] *

[1] IRIT, 31 400 Toulouse
[2] GREYC, 14 000 Caen

prenom.nom@irit.fr[1], prenom.nom@unicaen.fr[2]

Résumé

Dans cet article, nous étudions l'acquisition de GAI-décompositions de degré connu d'ordres de préférence dont un ensemble d'exemples est donné en entrée. Nous montrons que l'on peut représenter les GAI-décompositions cohérentes avec un ensemble d'exemples comme les solutions d'un système d'équations linéaires. Nous en déduisons un algorithme d'apprentissage passif (utilisant seulement des exemples observés) pour les GAI-décompositions de degré connu et constant. Nous montrons enfin comment généraliser ce résultat pour calculer des GAI-décompositions de degré ou de taille minimaux.

Abstract

We study acquisition of a GAI-decomposition of the preference order of the user through a set of examples representative of her preferences. We show that one can represent the GAI-decompositions consistent with a set of examples as the solutions of a system of linear equations. We derive a passive learning algorithm (using only observed examples) for GAI-decomposition of known and constant degree. We show how to generalize this result to calculate GAI-decompositions of minimal degree or minimal size.

*Ce travail est partiellement financé par l'ANR (projets LARDONS, ANR-2010-BLAN-0215, et BR4CP, ANR-11-BS02-008).

1 Introduction

Le développement de systèmes interactifs d'aide à la décision et de systèmes de recommandation a mis en évidence la nécessité de modèles capables d'utiliser les préférences d'un utilisateur pour l'orienter dans ses choix. La modélisation des préférences par le biais de formalismes de représentation compacts fait l'objet de travaux soutenus en intelligence artificielle depuis plus d'une quinzaine d'années [13, 14, 5, 6, 9]. Ces formalismes permettent l'expression de modèles suffisamment flexibles et riches pour décrire des comportements de décision complexes. Pour être intéressants en pratique, ces formalismes doivent de plus permettre l'élicitation des préférences de l'utilisateur, et ce en restant à un niveau admissible d'interaction. La configuration de produits combinatoires dans sa version *business to customer* [12] et la recherche à base de préférences (preference-based search ; voir par exemple [16]) constituent de bons exemples de ce type de problème de décision où les préférences de l'utilisateur ne sont pas connues *a priori* ; dans ce type d'applications, l'interaction avec l'utilisateur doit rester endessous de 0,25 s en instantané, sans que la session totale (la phase d'élicitation totale) ne dépasse la vingtaine de minutes — et ce, alors que l'objet à recommander à l'utilisateur est à rechercher dans un ensemble combinatoire de possibilités.

Lorsque les préférences sont qualitatives et structurées par des interactions « simples » entre attributs (notamment, lorsqu'elles sont séparables), on peut faire appel à des représentations de type CP-net [5, 4, 6] ; l'un des intérêts de ce modèle est qu'il permet d'apprendre facilement et rapidement, par le biais de quelques questions, les préférences de l'utilisateur [10, 11]. Malheureusement, il est des relations de préférence simples et intuitivement satis-

faisantes entre objets combinatoires qui ne peuvent pas être représentées par un CP-net. Le modèle des GAI-décompositions, et leurs représentations par des réseaux GAI [1, 9], présentent une alternative aux CP-nets, en proposant de représenter la relation de préférence de l'utilisateur (supposée complète et transitive) par une fonction d'utilité, dont on exploite les éventuelles propriétés de décomposabilité. Ces travaux utilisent la notion de d'indépendance additive généralisée (d'où le terme GAI, pour *Generalized Addditive Independence*), introduite par Fishburn [8], pour décomposer la fonction d'utilité sur l'ensemble des caractéristiques des objets en une somme de petites fonctions d'utilité ne portant que sur certains groupes d'attributs. Ces GAI-décompositions permettent donc d'exprimer des interactions générales entre les attributs, tout en préservant une certaine décomposabilité du modèle.

La question de l'élicitation de GAI-décompositions est au centre des travaux portant sur ce formalisme ; l'approche suivie jusqu'à présent suppose que la structure de la décomposition est donnée en entrée, ce qui est une hypothèse forte, que ne font par exemple pas les approches d'élicitation des préférences fondées sur des CP-nets.

Dans cet article, nous présentons une procédure d'apprentissage qui permet de construire une GAI-décomposition, aussi « décomposée » que possible, représentant un ensemble de préférences élémentaires données en exemple. Contrairement aux approches « classiques » de construction de GAI-décompositions, cette approche est une approche d'apprentissage passive,c'est-à-dire que l'on utilise un ensemble d'exemples fourni par un utilisateur mais qu'on ne s'autorise pas à poser des questions à l'utilisateur(contrairement à ce qui ce fait en apprentissage actif) pendant la phase d'apprentissage. Cette approche prend en entrée un ensemble d'exemples (des préférences élémentaires entres objets, c'est-à-dire des couples d'objets ordonnés par l'utilisateur), et retourne une GAI-décomposition minimale cohérente avec ces exemples. De plus on ne fait aucune hypothèse sur la structure des préférences (hormis, par définition, la complétude et la transitivité de la relation)

Le présent article est structuré comme suit. Après un rappel sur les GAI-décompositions et une présentation des principales notations (section 2), nous montrons en section 3 comment toute préférence élémentaire peut être représentée par une inéquation linéaire, dont les variables sont les éléments des tables d'une GAI-décomposition de degré k au maximum, l'idée étant de minimiser le degré de la décomposition apprise. La section 4 est dédiée au problème de l'apprentissage d'une GAI-décomposition proprement dit, du point de vue théorique (dimension de Vapnik-Chervonenkis) et algorithmique. Nous étendons cette approche au cas de l'apprentissage de décompositions minimales en section 5.

2 GAI-décompositions : principes et notations

Dans le formalisme GAI, la relation de préférence d'un utilisateur sur un ensemble d'objets χ est un préordre complet $\succeq$, c'est-à-dire qu'elle est complète et transitive. $o \succeq o'$ signifie que l'objet o est au moins aussi intéressant pour l'utilisateur que l'objet o'. Nous noterons $\succ$ la partie asymétrique de $\succeq$, et $\sim$ sa partie symétrique.

Le formalisme GAI propose de représenter de manière locale une relation de préférence sur un domaine combinatoire. Les objets de χ sont décrits par un ensemble de n variables $X = \{X_1, \ldots, X_n\}$, chaque X_i prenant ses valeurs sur un domaine D_i, généralement discret : les objets sont les éléments du produit cartésien des domaines des variables ($\chi = D_1 \times \cdots \times D_n$). Dans le présent article, pour faciliter la présentation, on considère des domaines booléens et on notera $D_i = \{x_i, \bar{x_i}\}$ le domaine de chaque variable X_i. Cette considération se fait sans perte de généralité. Les objets de χ sont des n-uplets, mais on pourra les noter, par abus de notation, comme des suites de valeurs ; ainsi pour $X = \{X_1, X_2, X_3, X_4\}$, l'objet $o = (x_1, \bar{x_2}, x_3, x_4)$ sera noté $x_1\bar{x_2}x_3x_4$. Enfin, pour tout sous-ensemble de variables $Z \subseteq X$, $o[Z]$ est la projection de o sur les variables de Z ; $\chi[Z] = \{o[Z], o \in D_1 \times \cdots \times D_n\}$ est l'ensemble des projections des éléments de χ sur Z (l'ensemble des affectations possibles des variables de Z).

Toute relation de préférence complète et transitive peut évidemment être représentée par une fonction d'utilité $u : \chi \mapsto \mathbb{R}$ vérifiant $o \succeq o' \Leftrightarrow u(o) \geq u(o')$ pour tous $o, o' \in \chi$. Cependant, l'ensemble χ étant combinatoire (il contient 2^n objets), il est impossible en pratique d'éliciter ou de mémoriser explicitement la relation $\succeq$, ou même la fonction u. Toutefois, dans certains cas, les préférences de l'utilisateur présentent une propriété d'indépendance forte entre attributs [7] ; il est alors possible de les représenter par un ensemble de fonctions d'utilité locales $u_i : D_i \mapsto \mathbb{R}$, d'arité 1 (donc peu coûteuses en mémoire et faciles à éliciter), et permettant de déduire directement l'utilité globale (et donc $\succeq$) : $u(o) = \Sigma_i u_i(o[X_i])$. La propriété correspondant à ces représentations est appelée l'*indépendance additive*. Malheureusement, les relations de préférence la satisfont rarement.

Exemple 1. *Soit* $X = \{X_1, X_2, X_3\}$ *avec* $D_1 = \{viande(v), poisson(p)\}$, $D_2 = \{rouge(r), blanc(b)\}$ *et*

$D_3 = \{citron(c), moutarde(m)\}$. *L'ensemble* χ *contient* 2^3*=8 objets possibles :*

$$o_1 = (v, r, c) \quad o_5 = (p, r, c)$$
$$o_2 = (v, r, m) \quad o_6 = (p, r, m)$$
$$o_3 = (v, b, c) \quad o_7 = (p, b, c)$$
$$o_4 = (v, b, m) \quad o_8 = (p, b, m)$$

Considérons maintenant l'ordre de préférence suivant :

$$o_2 \succ o_1 \succ o_6 \succ o_5 \sim o_4 \succ o_3 \succ o_8 \succ o_7$$

Dans ce premier exemple, la relation $\succeq$ *peut être représentée par une fonction d'utilité additive* $u(o) = \Sigma_i u_i(o[X_i])$*, en utilisant par exemple*

$$u_{\{X_1\}}(v) = 5, u_{\{X_1\}}(p) = 2$$
$$u_{\{X_2\}}(r) = 5, u_{\{X_2\}}(b) = 1$$
$$u_{\{X_3\}}(c) = 2, u_{\{X_3\}}(m) = 3$$

Exemple 2. *L'ordre de préférence suivant, sur les objets définis en exemple 1, constitue un second exemple, qui ne peut pas être représenté par une somme de fonction additive d'arité 1.*

$$o_2 \succ o_4 \sim o_7 \succ o_1 \succ o_8 \sim o_5 \succ o_3 \succ o_6$$

On voit ici que le menu $o_7 = (p, b, c)$ *est préféré au menu* $o_3 = (v, b, c)$*, et que le menu* $o_2 = (v, r, m)$ *est préférée au menu* $o_6 = (p, r, m)$ *; si les préférences étaient simplement additives, on en déduirait de la première comparaison que* $u_{\{X_1\}}(v) < u_{\{X_1\}}(p)$*, et de la seconde que* $u_{\{X_1\}}(v) > u_{\{X_1\}}(p)$*, ce qui constitue une contradiction.*

Pour représenter ce second exemple, on peut définir des fonctions d'utilité locales portant sur plusieurs *variables; la première* ($u_{\{X_1,X_2\}}$) *porte sur le plat principal* et *le vin, la seconde* (u_{X_1,X_3}) *porte sur le plat principal* et *l'accompagnement — la fonction d'utilité globale étant définie par* $u(o) = u_{\{X_1,X_2\}}(o[\{X_1, X_2\}]) + u_{\{X_1,X_3\}}(o[\{X_1, X_3\}])$ *:*

$$u_{\{X_1,X_2\}}(v, r) = 5, \qquad u_{\{X_1,X_2\}}(v, b) = 2$$
$$u_{\{X_1,X_2\}}(p, r) = 1, \qquad u_{\{X_1,X_2\}}(p, b) = 4$$
$$u_{\{X_1,X_3\}}(v, c) = 3, \qquad u_{\{X_1,X_3\}}(v, m) = 7$$
$$u_{\{X_1,X_3\}}(p, c) = 5, \qquad u_{\{X_1,X_3\}}(p, m) = 2$$

On voit dans l'exemple 1 que pour certaines relations de préférence, des variables peuvent être dépendantes d'autres, et que dans ce cas la fonction d'utilité est décomposée, non pas en utilités élémentaires, mais selon des fonctions d'utilité liant les variables dépendantes. Une telle décomposition de la fonction u est appelée une *GAI-décomposition* de u :

Définition 1 (GAI-décomposition)**.** *Soit* $X = \{X_1, ..., X_n\}$ *un ensemble de variables,* $\chi = D_1 \times \cdots \times D_n$ *un ensemble d'objets combinatoires et* $u : \chi \mapsto \mathbb{R}$ *une fonction d'utilité sur* χ*.*

Une GAI-décomposition *de* u *est un ensemble fini* $G = \{u_{Z_1}, \ldots, u_{Z_m}\}$ *de fonctions d'utilité portant sur des sous ensembles* Z_i *de* X *(c'est-à-dire* $u_{Z_i} : \chi[Z_i] \mapsto \mathbb{R}$*) et telles que l'on ait* $u(o) = \Sigma_{i=1,m}\ u_{Z_i}(o[Z_i])$ *pour tout* o *de* χ*.*

Les fonctions d'utilité locales sont également appelées *GAI-tables*, car on les implémente typiquement par des tables.

Il est clair que, via une fonction d'utilité, une GAI-décomposition G représente exactement une relation de préférence sur χ :

Définition 2 (relation représentée)**.** *Soit* $G = \{u_{Z_1}, \ldots, u_{Z_m}\}$ *une GAI-décomposition d'une fonction d'utilité* $u_G : \chi \mapsto \mathbb{R}$ *(c'est-à-dire* $u_G = \Sigma_{i=1,m} u_{Z_i}$*). On dit que* G représente *la relation de préférence* $\succeq$ *sur* χ *(ou simplement que c'en est une GAI-décomposition) si pour tous objets* $o, o' \in \chi$ *on a* $(o \succeq o' \iff u_G(o) \geq u_G(o'))$*.*

Dans la suite, on notera $\succeq_G$ la relation générée par un GAI G donné (c'est-à-dire $o \succeq_G o' \iff u_G(o) \geq u_G(o')$).

Définition 3 (degré)**.** *Le* degré *d'une GAI-décomposition* $G = \{u_{Z_1}, \ldots, u_{Z_m}\}$*, noté* $deg(G)$*, est le cardinal du plus grand* Z_i *de* G*.*

Bien entendu, $\{u\}$ est une GAI-décomposition (triviale) de u, de degré $Card(X)$. On peut aussi construire des GAI-décompositions non significatives en ajoutant des fonctions constantes $u_{Z_i} = 0$ à une GAI-décomposition G.

De façon plus intéressante, une même relation de préférence peut être représentée par plusieurs fonctions d'utilité, par exemple, parce que toute transformation affine (à coefficients positifs) d'une fonction d'utilité représentant un ordre représente le même ordre. D'autre part, une fonction d'utilité donnée admet plusieurs GAI décompositions. Dans l'ensemble des GAI-décompositions représentant $\succeq$, on s'intéressera tout particulièrement aux représentations minimales au sens de la décomposition.

Définition 4 (minimalité)**.** *Une GAI décomposition* $G = \{u_{Z_1}, \ldots, u_{Z_m}\}$ raffine *une autre GAI décomposition* $G' = \{u'_{Z'_1}, \ldots, u'_{Z'_{m'}}\}$ *si pour tout* Z_i*, il existe* $Z'_j \supseteq Z_i$*. Le raffinement est dit* strict *si l'inclusion est stricte pour au moins l'un des* Z_i*, ou s'il existe un* Z'_j *avec aucun* $Z_i \subseteq Z'_j$*. Une GAI-décomposition* G *d'une relation de préférence* $\succeq$ *est dite* minimale *si aucune autre GAI décomposition de* $\succeq$ *ne la raffine strictement.*

Ces notions sont résumées et illustrées dans l'exemple suivant.

Exemple 3. *Reprenons l'exemple 2 c'est à dire l'ordre de préférence suivant :*

$$o_2 \succ o_4 \sim o_7 \succ o_1 \succ o_8 \sim o_5 \succ o_3 \succ o_6$$

On peut représenter cette relation par la GAI-décomposition $G = \{u_{\{X_1,X_2,X_3\}}\}$ *comportant une seule fonction d'utilité* $u_{\{X_1,X_2,X_3\}}$ *dont la table est la suivante :*

v, r, c	8
v, r, m	12
v, b, c	5
v, b, m	9
p, r, c	6
p, r, m	3
p, b, c	9
p, b, m	6

Cette décomposition n'est pas minimale, puisqu'elle représente la même fonction d'utilité u que la décomposition $G' = \{u_{\{X_1,X_2\}}, u_{\{X_1,X_3\}}\}$ *donnée dans l'exemple 2.* G'*, elle, est minimale.*

3 Représentations par inégalités linéaires

Nous montrons dans cette section que l'ensemble des GAI-décompositions d'une relation de préférence donnée peut être caractérisé comme l'ensemble des solutions d'un système d'inégalités linéaires. Le nombre de variables du système dépend exponentiellement du degré des décompositions visées, mais polynomialement (à degré fixé) du nombre d'attributs utilisés pour décrire les objets du domaine.

Nous verrons dans la section 4 que cette construction permet d'apprendre des décompositions cohérentes avec un ensemble d'exemples donné en entrée.

Définition 5 (exemple)**.** *Soit* $\succeq$ *une relation de préférence sur* $\chi = X_1 \times \cdots \times X_n$*. Un* exemple *e de* $\succeq$ *est un triplet de la forme* (o, R, o')*, où* $o, o' \in \chi$ *sont des objets et R est dans* $\{\succ, \succeq, \sim, \preceq, \prec\}$*.*
Pour un ensemble *d'exemples E, on note* O_E *l'ensemble des objets qui interviennent dans au moins un exemple de E.*

Les exemples formalisent l'information reçue par l'apprenant, en particulier en observant l'utilisateur.

Définition 6 (cohérence)**.** *Soit E un ensemble d'exemples. Une fonction d'utilité u est dite* cohérente *avec E si pour tout exemple* $(o, R, o') \in E$*, on a* $u(o) > u(o')$ *(respectivement* $u(o) \geq u(o')$*, etc.) si R est la relation* $\succ$ *(respectivement* $\succeq$*, etc.).*

Dans la suite, on supposera que l'ensemble d'exemples est *sain*, c'est-à-dire qu'il existe une relation $\succeq^*$, complète et transitive, telle que pour tout exemple $(o, \succ, o') \in E$ (resp. tout exemple $(o, \succeq, o') \in E$), on ait $o \succ^* o'$ (resp. $o \succeq^* o'$). On utilisera ainsi la même notation, $\succeq$, pour la relation cible et pour sa restriction aux exemples.

Définition 7 (inégalité linéaire associée à un exemple)**.** *Soit* $e = (o, R, o')$ *un exemple d'une relation de préférence* $\succeq$ *sur* $\chi = X_1 \times \cdots \times X_n$*, et soit* $k \in \{1, \ldots, n\}$*. Soit* $\sigma > 0$ *une constante réelle, strictement positive mais arbitraire.*

Pour tout sous-ensemble de variables $Z \subseteq X$ *et toute affectation z de Z, on introduit une variable* U_z *à valeurs réelles.*

*L'*inégalité linéaire $inég_k(e)$ *associée à* $e = (o, R, o')$ *et k est définie comme*

$$\sum_{Z \subseteq X, 0 < |Z| \leq k} U_{o[Z]} \geq \sum_{Z \subseteq X, 0 < |Z| \leq k} U_{o'[Z]} + \sigma$$

si R est la relation $\succ$*, comme*

$$\sum_{Z \subseteq X, 0 < |Z| \leq k} U_{o[Z]} \geq \sum_{Z \subseteq X, 0 < |Z| \leq k} U_{o'[Z]}$$

si R est la relation $\succeq$*, et de même pour* $R = \sim, \preceq, \prec$*.*

Si E est un ensemble d'exemples de $\succeq$*, le* système linéaire *associé à E et k est défini comme le système d'inégalités linéaires* $\Sigma_k(E) = \bigwedge_{e \in E} inég_k(e)$*.*

Les variables $U_{o[z]}$ codent les éléments des GAI-tables de la décomposition G à apprendre : on définira G par $u_Z(z) = U_{o[Z]}$ pour $|Z| \leq k$ et $o \in O_E$, et par $u_Z(z) = 0$ sinon. D'autre part, nous utilisons une constante σ (un step) pour coder une inégalité stricte avec l'objectif d'utiliser la programmation linéaire (la proposition 1, çi dessous, montre que ceci est sans perte de généralité) : si l'on considérait des inégalités strictes (et, partant, l'ensemble de toutes les GAI-décompositions cohérentes avec E), nous considérerions un espace topologique ouvert.

Exemple 4. *Soient* $o = x_1 x_2 \bar{x_3}$ *et* $o' = \bar{x_1} x_2 x_3$*. L'inégalité linéaire associée à l'exemple* $e = (o, \succ, o')$ *pour* $k = 2$ *et* $\sigma = 0,1$ *est :*

$$\begin{aligned} & U_{x_1} + U_{x_2} + U_{\bar{x}_3} + U_{x_1 x_2} + U_{x_1 \bar{x_3}} + U_{x_2 \bar{x_3}} \\ \geq \; & U_{\bar{x_1}} + U_{x_2} + U_{x_3} + U_{\bar{x_1} x_2} + U_{\bar{x_1} x_3} + U_{x_2 x_3} + 0.1 \end{aligned}$$

Notons que le système $\Sigma_k(E)$ a au plus $\sum_{i=0}^{k} 2^i \binom{n}{i}$ inconnues (au plus autant que d'affectations possibles pour chaque sous-ensemble d'au plus k variables de X). En réalité, cependant, ce nombre de variables est moindre : la variable U_z n'apparaît que s'il existe un

objet $o \in O_E$ avec $o[Z] = z$. De fait, le nombre d'inconnues dans une inégalité est au plus $2.\sum_{i=0}^{k}\binom{n}{i}$, et le nombre total d'inconnues dans le système $\Sigma_k(E)$ est au plus $\sum_{i=0}^{k}\binom{n}{i}.|O_E|$.

Nous montrons maintenant que le système linéaire de la définition 7 caractérise les GAI-décompositions de degré au plus k cohérentes avec E. Pour des raisons techniques cependant, nous nous restreignons aux fonctions d'utilités *de step au moins* σ, c'est-à-dire aux fonctions u vérifiant $|u(o) - u(o')| \geq \sigma$ pour tous o, o' avec $u(o) \neq u(o')$. Ceci est sans perte de généralité en vertu de la proposition suivante (qui invoque essentiellement l'invariance de l'ordre induit par une transformation affine à coefficient positif).

Proposition 1. *Soit E un ensemble d'exemples et $\sigma > 0$ un réel arbitraire. Il existe une fonction d'utilité u et une GAI-décomposition de u de la forme $(u_{Z_1}, \ldots, u_{Z_m})$ et cohérente avec E si et seulement s'il existe une fonction u' avec une GAI-décomposition $(u'_{Z_1}, \ldots, u'_{Z_m})$ tels que u' soit cohérente avec E et de step au moins σ.*

Preuve Il suffit d'observer que si u est de step $\sigma_u < \sigma$, alors la fonction u' définie par la décomposition $((\sigma/\sigma_u)u_{Z_1}, \ldots, (\sigma/\sigma_u)u_{Z_m})$ représente la même relation de préférence que u, et est donc également cohérente avec E tout en ayant clairement un step supérieur ou égal à σ. □

Proposition 2. *Soit $\succeq$ une relation de préférence sur $\chi = X_1 \times \cdots \times X_n$, soit E un ensemble d'exemples de $\succeq$, soit $k \in \{1, \ldots, n\}$, et soit $\sigma > 0$ arbitraire. Alors les GAI-décompositions de degré au plus k, de step au moins σ, et cohérente avec E sont en bijection avec les solutions du système $\Sigma_k(E)$ (considéré comme possédant toutes les variables U_z, $|z| < k$, même si elles n'apparaissent pas dans les inégalités).*

Preuve La preuve est directe en utilisant, pour une GAI-décomposition telle que dans l'énoncé, l'affectation s des variables $U_{o[Z]}$ du système linéaire définie par $s(U_{o[Z_i]}) = u_{Z_i}(o[Z_i])$ pour tous $i \in \{1, \ldots, m\}, o \in \chi$. □

4 Apprentissage

Nous montrons dans cette section qu'il est possible d'apprendre de façon passive une GAI-décomposition d'une relation de préférence $\succeq^*$, en utilisant un nombre d'exemples ordinaux (de la forme (o, R, o')) qui croît exponentiellement avec le degré k de cette décomposition, mais qui, à k fixé, croît polynomialement avec le nombre d'attributs. En particulier, si l'on note G_k la classe de toutes les relation de préférence $\succeq$ qui ont une GAI décomposition de degré au plus k, alors la classe G_k est apprenable à partir d'exemples seulement.

Pour cela, nous donnons tout d'abord une borne théorique sur le nombre d'exemples nécessaires (section 4.1), puis nous donnons un algorithme pour le cas d'un degré k borné et connu (section 4.2). Dans les grandes lignes, l'algorithme maintient l'*espace des versions* de toutes les GAI-décompositions de degré k *et de step au moins* σ cohérentes avec les exemples reçus, en le représentant de façon compacte par le système linéaire $\Sigma_k(E)$. La VC-dimension de la classe G_k permet de conclure que l'algorithme converge vers $\succeq^*$ après réception d'un nombre polynomial d'exemples.

4.1 VC dimension

La *dimension de Vapnik-Chervonenkis* (VC-dimension) d'une classe de concepts C mesure en un certain sens l'expressivité de C et, partant, la difficulté d'apprendre des concepts de C à partir d'exemples. Intuitivement, la VC-dimension de C mesure le plus grand nombre d'exemples « indépendants » les uns des autres, en ce sens que leurs étiquettes ne dépendent d'aucune manière les unes des autres. Plus formellement, nous utiliserons les deux définitions suivantes, spécialisées pour notre contexte (pour rendre les définitions concrètes, voir par exemple l'appartenance de (o, o') au concept c, c'est-à-dire $c(o, o') = 1$, comme la validité de l'assertion $o \succeq o'$).

Définition 8 (éclaté)**.** *Soit C un ensemble de concepts binaires $c : \chi \times \chi \to \{0, 1\}$. Un sous-ensemble de couples d'objets $O \subseteq \chi \times \chi$ est dit* éclaté *par C si pour toute partition $\{O^1, O^0\}$ de O, il existe un concept $c \in C$ vérifiant $\forall (o, o') \in O^1, c(o, o') = 1$ et $\forall (o, o') \in O^0, c(o, o') = 0$.*

Définition 9 (VC-dimension)**.** *Soit C un ensemble de concepts binaires. La* dimension de Vapnik-Chervonenkis *(VC-dimension) de C est la taille du plus grand ensemble d'objets $O \subseteq \chi \times \chi$ qui est éclaté par C.*

Même s'il existe des résultats plus généraux, les résultats d'apprenabilité (ou de non-apprenabilité) en fonction de la VC-dimension concernent la VC-dimension de concepts binaires.

Nous voyons donc une relation de préférence $\succeq$ comme deux relations, notées $\succ$ et $\prec$. Formellement, la relation $\succ$ associée à une relation de préférence $\succeq$ est le concept binaire $C_\succ \subseteq \chi \times \chi$ défini par $C_\succ(o, o') = 1$ pour $o \succeq o'$ et $o' \not\succeq o$, et $C_\succ(o, o') = 0$ sinon (c'est-à-dire si l'on a $o \preceq o'$), et le concept $C_\prec$ est défini de manière duale.

Proposition 3. *Soit $C_\succ^{n,k}$ la classe des concepts binaires de la forme $C_\succ$ où $\succeq$ est une relation de*

préférence sur $\chi = X_1 \times \cdots \times X_n$ représentable par une GAI-décomposition de degré k. La VC-dimension pour des attributs binaires de $C_\succ^{n,k}$ est en $O(2^k n^{k+1})$.

Plus généralement, si les domaines des variables ne sont pas booléens alors la VC-dimension est en $O(|D|^k n^{k+1})$ où D est le domaine des variables. **Preuve** Soit $K = \sum_{i=0}^{k} 2^i \binom{n}{i}$ (et donc, $K \in O(2^k n^{k+1})$). Nous montrons qu'aucun ensemble de couples d'objets $O \subseteq \chi \times \chi$ de taille strictement supérieure à K n'est éclaté par $C_\succ^{n,k}$, ce qui concluera.

Soit donc $O \subseteq \chi \times \chi$ un ensemble de couples d'objets de taille au moins $K+1$. Pour chaque couple $(o, o') \in O$, dans l'esprit de la définition 7 nous définissons la somme (combinaison linéaire) formelle

$$V_{o,o'}^k = \sum_{Z \subseteq X, 0<|Z|\leq k} U_{o[Z]} - U_{o'[Z]}$$

qui correspond donc à la combinaison des membres gauche et droit de toute inégalité linéaire associée à o et o'.

Tous les $V_{o,o'}^k$ (pour $(o, o') \in O$) portent sur les mêmes K variables. Donc, si O contient au moins $K+1$ couples, il existe au moins un couple (ω, ω') tel que le vecteur $V_{\omega,\omega'}^k$ est une combinaison linéaire des autres vecteurs, c'est-à-dire qu'il existe $\lambda_{o,o'}$ (pour $(o, o') \in O \setminus \{(\omega, \omega')\}$) vérifiant

$$V_{\omega,\omega'}^k = \sum_{(o,o') \in O \setminus \{(\omega,\omega')\}} \lambda_{o,o'} V_{o,o'}^k$$

Soit O^- (resp. O^+) le sous-ensemble de $O \setminus \{(\omega, \omega')\}$ constitué des couples (o, o') avec $\lambda_{o,o'} \leq 0$ (resp. $\lambda_{o,o'} > 0$).

Si $O^+ \neq \emptyset$, nous montrons alors qu'aucun concept $c \in C_\succ^{n,k}$ n'est cohérent avec la partition $O^1 = O^+, O^0 = O^- \cup \{(\omega, \omega')\}$, c'est-à-dire avec les étiquettes $o \succ o'$ pour tout $(o, o') \in O^+$, $o \preceq o'$ pour tout $(o, o') \in O^-$, et $\omega \preceq \omega'$. Dans le cas $O^+ = \emptyset$, on a il est facile de voir que le système implique $\omega \preceq \omega'$, et c'est alors $C_\succ(\omega, \omega') = 1$ qui est impossible.

En utilisant les propositions 1 et 2 nous obtenons immédiatement que le système linéaire suivant doit avoir une solution (pour un $\sigma > 0$ arbitraire) pour qu'il existe une fonction d'utilité u ayant une GAI-décomposition de degré k et consistante avec les étiquettes susmentionnées :

$$\begin{aligned} V_{o,o'}^k &\geq \sigma \quad (\forall (o, o') \in O^+) \\ V_{o,o'}^k &\leq 0 \quad (\forall (o, o') \in O^-) \end{aligned}$$

Du fait des signes des $\lambda_{o,o'}$, il s'ensuit que les équations suivantes doivent être vérifiées :

$$\begin{aligned} \lambda_{o,o'} V_{o,o'}^k &\geq \lambda_{o,o'} \sigma \quad (\forall (o, o') \in O^+) \\ \lambda_{o,o'} V_{o,o'}^k &\geq 0 \quad (\forall (o, o') \in O^-) \end{aligned}$$

Toute solution de ce système doit donc satisfaire

$$\sum_{(o,o') \in O \setminus \{(\omega,\omega')} \lambda_{o,o'} V_{o,o'}^k \geq \sigma \sum_{(o,o') \in O^+} \lambda_{o,o'}$$

c'est-à-dire

$$V_{\omega,\omega'} \geq \sigma \sum_{(o,o') \in O^+} \lambda_{o,o'}$$

En utilisant $O^+ \neq \emptyset$ et à nouveau la proposition 2, nous obtenons $\omega > \omega'$, donc $C_\succ(\omega, \omega') = 0$ est impossible et donc, O n'est pas éclaté par $C_\succ^{n,k}$.

Si par contre $O^+ = \emptyset$, nous montrons qu'aucun concept $c \in C_\succ^{n,k}$ n'est cohérent avec une partition (O^1, O^0) où O^1 contient $O^- \cup \{(\omega, \omega')\}$: en effet si peut avoir $O^- \subseteq O^1$, alors les $V_{o,o'}^k$ pour $(o, o') \in O^-$ sont > 0, et donc

$$V_{\omega,\omega'}^k = \sum_{(o,o') \in O^-} \lambda_{o,o'} V_{o,o'}^k \leq 0$$

et donc forcément $(\omega, \omega') \in O^0$.

Dans les deux cas, on conclut que O n'est pas éclaté par $C_\succ^{n,k}$, et puisque O est arbitraire de taille $K+1$, que la VC-dimension de $C_\succ^{n,k}$ est au plus K.

□

4.2 Algorithme

Nous décrivons maintenant l'algorithme permettant d'apprendre une GAI-décomposition cohérente avec un ensemble d'exemples donnés. Afin de laisser la présentation claire, nous nous plaçons dans le cadre du *PAC-apprentissage* (« probablement approximativement correct ») introduit par Valiant [15].

Nous nous plaçons dans un cadre d'apprentissage passif : l'apprenant reçoit un nombre m d'exemples (o, R, o') du concept cible $\succeq^*$, puis doit calculer un concept $\succeq$. Le nombre m d'exemples demandés est choisi par l'apprenant en fonction du nombre de variables n et de deux paramètres $\varepsilon, \delta \in]0, 1[$. Chaque exemple est tiré aléatoirement, indépendamment et avec remise, selon une distribution de probabilité p sur $\chi \times \chi$ qui est fixe mais inconnue de l'apprenant. Pour o, o' ainsi tirés, l'exemple (o, R, o') est communiqué à l'apprenant, où R est déterminé sans bruit par le concept cible $\succeq^*$. Un *PAC*-apprenant est un algorithme qui possède les propriétés suivantes :

- avec probabilité au moins $1-\delta$, il retourne un concept $\succeq$ qui est ε-correct, c'est-à-dire que la masse, selon la distribution de probabilité p utilisée pour générer les exemples, des couples (o, o') incorrectement classés par $\succeq$ est au plus ε ; formellement, $\sum\{p(o, o') \mid oRo' \text{ mais } \neg(o \; R^* \; o')\} \leq \varepsilon$, où R est une relation quelconque dans $\{\succeq, \succ, \sim, \prec, \preceq\}$,

– le nombre m d'exemples demandé par l'algorithme est polynomial en $n, 1/\varepsilon, 1/\delta$,
– le temps de calcul de l'algorithme est polynomial en $n, 1/\varepsilon, 1/\delta$, en comptant pour une unité la demande et la réception d'un exemple.

Une classe de concepts est dite *PAC-apprenable* s'il existe un PAC-apprenant pour elle.

Intuitivement, ce cadre formalise les situations dans lesquelles l'apprenant observe pendant un certain temps des objets de son environnement (ceux qui se présentent, sans qu'il ne puisse les choisir), et est aidé par un « enseignant », qui lui fournit pour chaque objet rencontré l'étiquette correcte. La distribution de probabilité p permet de formaliser des situations dans lesquelles tous les objets ne sont pas aussi courants les uns que les autres dans l'environnement. Ainsi, un objet rare aura peu de chances d'être rencontré pendant la phase d'apprentissage (probabilité faible selon p), mais en contrepartie, l'apprenant sera peu pénalisé pour une erreur sur un tel objet.

Pour montrer que les GAI-décompositions de degré k connu et borné sont PAC-apprenables, nous utilisons une approche standard, connue sous le nom d'« apprentissage cohérent » (*consistent learning*). Dans cette approche, l'apprenant maintient à tout instant un concept cohérent avec tous les exemples reçus jusqu'alors ; en l'occurrence, notre apprenant les maintient essentiellement tous (au *step* σ près) en intention, via le système $\Sigma_k(E)$. Des résultats génériques permettent de conclure sur le nombre m d'exemples nécessaires en utilisant la VC-dimension.

L'algorithme est représenté ci-dessous. La valeur de m est donnée plus loin (Proposition 4). Notons que les exemples étant sans bruit, l'approche par consistance retourne toujours une solution, en l'occurrence, le système $\Sigma_k(E)$ a nécessairement une solution.

Algorithm 1 GAI-Learning

Require: $k < n$
1: $\Sigma_k(E) = \emptyset$
2: **for** $i = 1, \ldots, m$ **do**
3: demander un exemple e de la forme (o, R, o')
4: Ajouter $in\acute{e}g_k(e)$ à $\Sigma_k(E)$
5: **end for**
6: résoudre $\Sigma_k(E)$
7: **return** retourner une GAI-décomposition construite à partir d'une solution de $\Sigma_k(E)$

Proposition 4. *Pour toute constante $k > 0$, la classe C^k des GAI-décompositions de degré au plus k est PAC-apprenable. Le nombre m d'exemples requis par l'algorithme* GAI-Learning *est en* $O(\max(\frac{1}{\varepsilon}\log\frac{1}{\delta}, \frac{2^k n^{k+1}}{\varepsilon}\log\frac{1}{\varepsilon}))$.

Preuve La proposition 2 montre que la GAI décomposition retournée par l'algorithme est cohérente avec l'ensemble de tous les exemples reçus.

Une conséquence directe d'un résultat générique de Blumer et al. [3, Theorem 2.1 (ii)] est qu'un nombre d'exemples m en $O(\max(\frac{1}{\varepsilon}\log\frac{1}{\delta}, \frac{2^k n^k}{\varepsilon}\log\frac{1}{\varepsilon}))$ suffit pour apprendre une GAI-décomposition G telle que l'ordre associé $\succeq_G$ soit ε-correct (même si la VC-dimension telle que donnée dans la proposition 3 concerne les concepts binaires $C_\succ$ et $C_\prec$, les mêmes exemples sont utilisés pour apprendre à la fois le concept binaire $C_\succ$ et le concept binaire $C_\prec$, via le système $\Sigma_k(E)$; pour plus de détails sur ce type de constructions, nous renvoyons le lecteur à [2]).

En outre, m étant polynomial en $n, 1/\epsilon, 1/\delta$ (rappelons que k est considéré comme constant), le système $\Sigma_k(E)$ est de taille polynomiale, et en tant que système linéaire, il peut être résolu en temps polynomial.

□

5 Apprentissage d'une décomposition minimale

Dans la section précédente, nous avons considéré le problème d'apprentissage d'une GAI-décomposition d'une fonction d'utilité cible, sans autre restriction qu'une borne sur le degré maximal k, connue à l'avance.

Nous montrons maintenant comment généraliser ce résultat en calculant des GAI-décompositions aussi « simples » que possible. Pour cela, nous proposons différents problèmes d'optimisation, basés sur le système linéaire $\Sigma_k(E)$. Nous explorons deux notions de « simplicité » d'une GAI-décomposition, toujours en supposant une borne k, connue, sur le degré de la décomposition cible.

Nous cherchons tout d'abord à maximiser le nombre d'entrées nulles dans les tables de la GAI-décomposition apprise, c'est-à-dire à maximiser le nombre de couples (Z, z) avec $u_Z(z) = 0$. Optimiser cette mesure permet en particulier de réduire l'espace nécessaire pour stocker la décomposition apprise.

Pour atteindre cet objectif, étant donné un degré k fixé et un ensemble d'exemples E, on définit le problème d'optimisation linéaire suivant :

$$\text{(P1)} \begin{cases} \text{minimiser} \sum_{Z\subseteq X, 0<|Z|\leq k, o\in O_E} U_{o[Z]} \\ \text{sous les contraintes} \\ \bullet\ in\acute{e}g_k(e) \text{ pour tout } e \in E \\ \bullet\ U_{o[Z]} \geq 0 \text{ pour tous } Z, o \end{cases}$$

Bien que (P1) permette d ?obtenir une certaine forme de minimalité, il ne suffit pas à minimiser le nombre de coefficients non nuls dans les tables, comme le montre l'exemple suivant.

Exemple 5. *Soient deux variables booléennes X_1, X_2, et $E = \{x_1x_2 \succ x_1\bar{x}_2\ ,\ x_1\bar{x}_2 \succ \bar{x}_1\bar{x}_2\ ,\ \bar{x}_1\bar{x}_2 \succ \bar{x}_1x_2\}$. On fixe $k = 2$, $\sigma = 1$, et on considère les décompositions (u_1, u_{12}) et (u'_{12}), toutes deux consistantes avec E :*

$$u_1 : \begin{array}{|c|c|}\hline x_1 & 1 \\ \bar{x}_1 & 0 \\ \hline\end{array} \quad u_{12} : \begin{array}{|c|c|}\hline x_1x_2 & 2 \\ x_1\bar{x}_2 & 1 \\ \bar{x}_1\bar{x}_2 & 1 \\ \bar{x}_1x_2 & 0 \\ \hline\end{array} \quad \Big| \quad u'_{12} : \begin{array}{|c|c|}\hline x_1x_2 & 3 \\ x_1\bar{x}_2 & 2 \\ \bar{x}_1\bar{x}_2 & 1 \\ \bar{x}_1x_2 & 0 \\ \hline\end{array}$$

La décomposition (u'_{12}) a moins de coefficients non nuls que (u_1, u_{12}), pourtant cette dernière a une meilleure valeur dans (P1).

Une solution pour minimiser le nombre de coefficients non nuls consiste alors à effectuer un post-traitement peu coûteux aux solutions de (P1) : on reporte les valeurs des coefficients des tables de certains Z_i vers les tables de $Z_j \supset Z_i$.

Un autre objectif naturel est de minimiser le degré de la GAI-décomposition apprise (sous la borne k). Notons tout d'abord que cet objectif n'est pas réalisé, dans le cas général, par l'objectif précédent, autrement dit, que trouver un optimum des problèmes (P1) ne garantit pas qu'on a une GAI-décomposition de degré minimal :

Exemple 6. *Soient $X = \{X_1, X_2\}$, $D_1 = \{x_1, \bar{x}_1\}$ et $D_2 = \{x_2, \bar{x}_2\}$. Soit $E = \{x_1\bar{x}_2 \succ \bar{x}_1x_2, x_1x_2 \succ \bar{x}_1x_2, x_1\bar{x}_2 \succ x_1x_2\}$, donc $O_E = \{x_1\bar{x}_2, \bar{x}_1x_2, x_1x_2\}$ En cherchant une GAI-décomposition de degré au plus 2, on a le problème d'optimisation suivant :*

$$\begin{cases} \textit{minimiser} \displaystyle\sum_{Z\subseteq X, 0<|Z|\leq k, o\in O_E} U_{o[Z]} \\ \textit{sous les contraintes} \\ \bullet\ U_{x_1} + U_{\bar{x}_2} + U_{x_1\bar{x}_2} \geq U_{\bar{x}_1} + U_{x_2} + U_{\bar{x}_1x_2} + \sigma \\ \bullet\ U_{x_1} + U_{x_1x_2} \geq U_{\bar{x}_1} + U_{\bar{x}_1x_2} + \sigma \\ \bullet\ U_{\bar{x}_2} + U_{x_1\bar{x}_2} \geq U_{x_2} + U_{x_1x_2} + \sigma \\ \bullet\ U_{o[Z]} \geq 0 \textit{ pour tous } Z, o \end{cases}$$

En prenant par exemple $\sigma = 0.1$, on obtient comme solution optimale $U_{x_1x_2} = 0,1$, $U_{x_1\bar{x}_2} = 0.2$, toutes les autres valeurs étant nulles. On a donc une décomposition de degré 2, alors qu'il en existe une de degré 1, définie par $U_{x_1} = U_{\bar{x}_2} = 0.1$, toutes les autres valeurs étant nulles (d'après les exemples, les deux variables X_1 et X_2 sont indépendantes, avec x_1 comme valeur préférée pour X_1 et $\bar{x}_2$ pour X_2).

En pondérant les $U_{o[Z]}$ dans la fonction objectif, on peut pousser la minimisation et donc la mise à zéro de certains d'entre eux, et donc favoriser *a contrario* certaines cliques – les plus petites – dans la décomposition. On introduit donc des poids w_i, où chaque w_i pondérera, dans la fonction objectifs, les utilités de degré i (i.e. les $U_{o[Z]}$ tels que $|Z| = i$) . Formellement, on cherche à minimiser la fonction objectif suivante :

$$\sum_{Z\subseteq X \& |Z|\leq k \& o\in O_E} w_{|Z|} \times U_{o[Z]}$$

On doit fixer les w_is de manière à rendre un $U_{o[Z]}$ de degré i plus coûteux que l'ensemble des $U_{o[Z']}$ pour $Z' \subset Z$ et $|Z| = i - 1$; les w_is doivent alors vérifier

$$w_i \ \times\ \min(U_{o[Z]}) > i \ \times\ 2^{i-1} \ \times\ w_{i-1} \ \times\ \max(U_{o[Z]})$$

où le $\min(U_{o[Z]})$ est pris sur l'ensemble des $U_{o[Z]}$ non nuls. On peut imposer $U_{o[Z]} \leq 1$ pour tous les o, Z, ce qui conduit à la constante σ suffisamment petite pour ne pas rendre le systèmes d'équations linéaires artificiellement inconsistant. Pour imposer que les $U_{o[Z]}$ non nuls aient un minimum non nul, on peut introduire de nouvelles variables booléennes $V_{o[Z]}$, on alors le problème de programmation linéaire mixte suivant :

$$\text{(P2)} \begin{cases} \text{minimiser } (\sum_{Z\subseteq X} \text{ t.q. } {}_{|Z|\leq k, o\in O_E} w_{|Z|} \times U_{o[Z]}) \\ \text{sous les contraintes} \\ \text{1) } \textit{inég}_k(e) \text{ pour tout } e \in E \\ \text{2) } \alpha\ .\ V_{o[Z]} \leq U_{o[Z]} \leq V_{o[Z]} \\ \qquad\qquad \text{pour tout } Z \subseteq X, o \in O_E \\ \text{3) } V_{o[Z]} \in \{0, 1\} \end{cases}$$

Puisque $V_{o[Z]} \in \{0, 1\}$, les contraintes 2) imposent $U_{o[Z]} \in \{0\} \cup [\alpha, 1]$. Le minimum des $U_{o[Z]}$ pris sur l'ensemble des $U_{o[Z]}$ non nuls est alors α, et il suffit d'avoir $w_i > (i2^{i-1}/\alpha)w_{i-1}$ pour avoir un $U_{o[Z]}$ de degré i plus coûteux que l'ensemble des $U_{o[Z']}$ pour $Z' \subset Z$ et $|Z| = i - 1$. Il faut choisir la constante σ qui apparaît dans certaines des $\textit{inég}_k(e)$ suffisamment petite afin que la restriction à l'intervalle $[0, 1]$ des valeurs possibles pour les $U_{o[Z]}$ ne risque pas de supprimer artificiellement certaines solutions du problème.

Une autre solution consiste à minimiser directement la somme des V_z pondérés par les w_is afin d'apprendre une GAI décomposition minimale au sens de la décomposition (définition 4). Ici V_z représente une GAI-table de Z et w_i représente le coût de la création d'une table de degré i. On a alors le programme linéaire mixte suivant, en choisissant les w_is tels que $w_k = k{*}w_{k-1}{+}1$ avec $w_1 = 1$:

$$\text{(P3)} \begin{cases} \text{minimiser } (\sum_{Z \subseteq X} \text{ t.q. } _{|Z| \leq k} w_{|Z|} \times V_Z) \\ \text{sous les contraintes} \\ \bullet\ in\acute{e}g_k(e) \text{ pour tout } e \in E \\ \bullet\ V_Z \geq U_{o[Z]} \geq 0 \\ \qquad\qquad \text{pour tout } Z \subseteq X, o \in O_E \\ \bullet\ V_Z \in \{0,1\} \end{cases}$$

6 Conclusion

Nous avons montré que tout préordre complet [1] sur un ensemble d'objets combinatoires représentable par une GAI-décomposition de degré k peut également être encodé par un système d'inégalités linéaires, dont les solutions correspondent essentiellement aux GAI-décompositions de la relation initiale. Le principe de notre algorithme est de générer, étant donnés un ensemble d'exemples et un degré k fixé, un système linéaire, dont les variables sont les valeurs des GAI-tables recherchées. Le choix d'une fonction objectif pertinente permet alors d'utiliser un algorithme de programmation linéaire (éventuellement mixte) pour générer une GAI-décomposition qui soit de degré ou de taille minimaux.

Nous avons supposé que toutes les variables sont binaires, mais les résultats se généralisent aisément au cas où on a des variables dont les domaines ont au plus d valeurs, pour un certain d fixé : on vérifie que la VC-dimension de la classe de GAI correspondante est bornée par $d^k n^{k+1}$.

Dans la version de l'algorithme que nous proposons dans cet article, nous cherchons à couvrir tous les exemples. Si cela n'est pas possible avec le degré choisi, ou si l'ensemble d'exemples n'est pas sain, le système linéaire généré n'a pas de solution. Il est simple de relaxer cette exigence en introduisant dans les parties gauches des inégalités des variables de relaxation δ, dont la somme est à minimiser (la version courante correspondant à des δ nuls).

Le second développement nécessaire de ce travail consiste à établir une stratégie intelligente de choix du degré du GAI à apprendre. On peut procéder de manière croissante (supposer les variables indépendantes, puis augmenter incrémentalement le degré jusqu'à obtenir une bonne couverture des exemples). Une stratégie plus fine consisterait à analyser le graphe appris pour $k = 2$, afin d'augmenter notre connaissance sur les dépendances de variables (une clique sur un ensemble de variables Y induisant la nécessité d'une utilité locale u_Y). On pourrait plus généralement faire collaborer une procédure d'apprentissage (actif ou passif) du degré du GAI, ou mieux, de sa structure, et une procédure d'apprentissage des tables fondées sur leur représentation par un système d'équations linéaires.

Enfin, nous envisageons un retour sur les méthodes d'élicitation du GAI : il s'agirait de proposer une méthode, par exemple en utilisant un ensemble de questions à poser à l'utilisateur, qui permettrait de déduire les (in)dépendances entre variables. Dans ce contexte, les variables de notre système linéaire seraient introduites au fur et à mesure.

1. Si la relation de préférence de l'utilisateur n'est pas complète, on obtient un GAI représentant une préférence qui est compatible (en fait, qui raffine) avec la préférence original. Cela dit , si l'on souhaite travailler sur des préférences incomplète, il faut mieux utiliser les CP-net plutôt que les GAI

Références

[1] Fahiem Bacchus and Adam Grove. Graphical models for preference and utility. In *Proceedings of UAI'95*, pages 3–10, 1995.

[2] Shai Ben-David, Nicolò Cesa-Bianchi, David Haussler, and Philip M. Long. Characterizations of learnability for classes of $\{0, \ldots, n\}$-valued functions. *Journal Of Computer and System Sciences*, 50 :54–86, 1995.

[3] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the vapnik-chervonenkis dimension. *Journal Of ACM*, 36(4) :929–965, 1989.

[4] Craig Boutilier, Fahiem Bacchus, and Ronen I. Brafman. UCP-networks : A directed graphical representation of conditional utilities. In *Proceedings of UAI'01*, pages 56–64, 2001.

[5] Craig Boutilier, Ronen Brafman, Holger Hoos, and David Poole. Reasoning with conditional ceteris paribus preference statem. In *Proceedings of UAI-99*, pages 71–80, 1999.

[6] Craig Boutilier, Ronen I. Brafman, Carmel Domshlak, Holger H. Hoos, and David Poole. CP-nets : A tool for representing and reasoning with conditional ceteris paribus preference statements. *JAIR*, 21 :135–191, 2004.

[7] Gérard Debreu. Topological methods in cardinal utility theory. In K.J. Arrow, S. Karlin, and P. Suppes, editors, *Mathematical Methods in the Social Sciences*, pages 16–26. Stanford University Press, Stanford, 1960.

[8] Peter.C. Fishburn. *Utility Theory for Decision Making.* Wiley, New York, 1970.

[9] Christophe Gonzales and Patrice Perny. GAI networks for utility elicitation. In *Proceedings of KR'04*, pages 224–234, 2004.

[10] Frédéric Koriche and Bruno Zanuttini. Apprentissage de réseaux de préférences ceteris paribus.

In Laurence Cholvy and Sébastien Konieczny, editors, *Actes des Journées d'Intelligence Artificielle Fondamentale (IAF 2008)*, 2008.

[11] Frédéric Koriche and Bruno Zanuttini. Learning conditional preference networks with queries. In *Proceedings of IJCAI'09*, pages 1930–1935, 2009.

[12] Daniel Mailharro. A classification and constraint-based framework for configuration. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, pages 383–397, September 1998.

[13] Anand S. Rao and Michael P. Georgeff. Modeling rational agents within a BDI-architecture. In *Proceedings of KR'92*, pages 473–484, 1991.

[14] Thomas Schiex, Hélène Fargier, and Gérard Verfaillie. Valued constraint satisfaction problems : Hard and easy problems . In *Proceedings of IJCAI'95)*, pages 631–637. Morgan Kaufmann, août 1995.

[15] Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11) :1134–1142, 1984.

[16] Paolo Viappiani, Boi Faltings, and Pearl Pu. Preference-based search using example-critiquing with suggestions. *JAIR*, 27 :465–503, 2006.